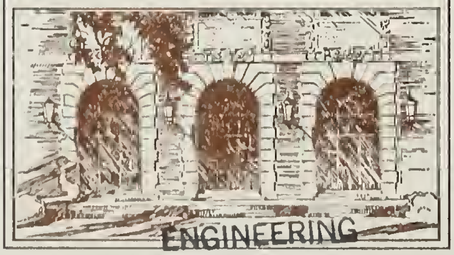


Center for Advanced Computation

UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN
URBANA, ILLINOIS 61801

CAC Document No. 21

A JACOBI ALGORITHM FOR ILLIAC IV

by

Lawrence M. McDaniel

November 8, 1971
(revised June 7, 1972)

CAC Document No. 21

A JACOBI ALGORITHM FOR ILLIAC IV

by

Lawrence M. McDaniel

Center for Advanced Computation
University of Illinois at Urbana-Champaign
Urbana, Illinois 61801

November 8, 1971
(revised June 7, 1972)

This work was supported in part by the Advanced Research Projects Agency of the Department of Defense and was monitored by the U. S. Army Research Office-Durham under Contract No. DAHC04-72-C-0001.

ABSTRACT

This revised version of CAC Document #21 supercedes the document dated November 8, 1971.

Several methods have been proposed to enable the computation of eigenvalues and eigenvectors of large, real symmetric or complex Hermitian matrices on ILLIAC IV.

One of the most effective methods in the utilization of parallel computations has proven to be a modified Jacobi algorithm. This document presents yet another modification which exploits the parallelism of ILLIAC IV to a greater extent than has been previously done.

Flow charts and the assembly language (ASK) routine JACOBI are included in the report.

ACKNOWLEDGEMENT

The author would like to thank Dr. Ahmed H. Sameh, who developed the modified Jacobi algorithm and provided guidance in its implementation.

TABLE OF CONTENTS

1.0 Introduction

The computation of eigenvalues and eigenvectors of large, real symmetric matrices is of great practical value in many fields. One of the most effective methods that has been examined to solve this problem on the ILLIAC IV is a parallel version of Jacobi's algorithm.

The main difference between this code and the one previously written [1] is that within each sweep, defined by (2m-1) transformations (where $m = [(n+1)/2]$, i.e. the greatest integer less than or equal to $(n+1)/2$, in which n is the order of the matrix), each orthogonal transformation annihilates different $[\frac{n}{2}]$ off-diagonal elements. This proved to be a substantial improvement that led to a greater speed of convergence.

2.0 Jacobi's Method

In the classical method of Jacobi, a real symmetric matrix is reduced to the diagonal form by a sequence of plane rotations $A_k = R_k A_{k-1} R_k^t$ $(k = 1,2...)$, where $A_o = A$ is the original matrix, and the rotation matrix R_k eliminates the off-diagonal element $a_{pq}^{(k)}$ (hence $a_{pq}^{(k)}$) through an angle $\alpha_{pq}^{(k)}$ [5]. See Appendix A for the appropriate value of $\alpha_{pq}^{(k)}$ to annihilate the element $a_{pq}^{(k)}$.

3.0 Modifications to the Classical Jacobi Method

3.1 Decomposition Into Block Diagonal Submatrices

Rather than searching for the largest off-diagonal element of A_{k-1} in the (p,q) position and eliminating a_{pq} and a_{qp}, A. Sameh and L. Han.[4] proposed a modified Jacobi algorithm where all off-diagonal elements of each 2 x 2 submatrix along the diagonal are eliminated through an orthogonal transformation.

In order to bring to the diagonal new submatrices with non-zero off-diagonal elements, the method necessitates a large degree of row and column shuffling which tends to be expensive on a parallel computer such as ILLIAC IV.

3.2 Optimal Construction of Orthogonal Transformations

A. Sameh[3] showed that a judicious choice of the pairs (p,q) can produce a modified Jacobi algorithm that attains maximum efficiency of parallel computation.

For example, for a matrix A of order 4, if the orthogonal transformation R is chosen as,

$$(3.1) \qquad R = \begin{bmatrix} c_1 & 0 & s_1 & 0 \\ 0 & c_2 & 0 & s_2 \\ -s_1 & 0 & c_1 & 0 \\ 0 & -s_2 & 0 & c_2 \end{bmatrix}$$

where $c_i = \cos \alpha_i (i = 1,2)$, then RAR^t would have zero elements in positions (1,3) and (2,4) provided that the angles α_1 and α_2 are properly chosen.

Define m= [(n+1)/2] where n is the order of the matrix and [ ] is the greatest integer function. Let each of the (2m-1) orthogonal transformations be denoted by a sweep. Noting that the maximum number of the $(n^2-n)/2$ off-diagonal elements which can be eliminated by an orthogonal transformation of the type (3.1) is [n/2], an optimal algorithm requires that, [3]:

1) Each orthogonal transformation R_k should eliminate [n/2] off-diagonal elements.

2) Each sweep should eliminate each off-diagonal element once, i.e. each of the 2m-1 orthogonal transformations in a sweep should annihilate different [n/2] off-diagonal elements.

3.2.1 Elimination Scheme

In [3] several schemes were proposed to satisfy the requirements discussed in the previous section. The scheme implemented in the JACOBI algorithm is described below:

For a given sweep, each of the (2m-1) orthogonal matrices R_k consist of the elements,

$$(3.2)\qquad R_{pp}^{(k)} = R_{qq}^{(k)} = \cos \alpha_{pq}^{(k)};\quad R_{pq}^{(k)} = -R_{qp}^{(k)} \begin{cases} = \sin \alpha_{pq}^{(k)}, & p < q, \\ = -\sin \alpha_{pq}^{(k)} & p > q, \end{cases}$$

where p and q are sequences defined by

(a) for $k = 1,2,\ldots,m-1,$

$$q = m-k+1,\ m-k+2,\ \ldots,\ n-k.$$

$$(3.3)\qquad \begin{aligned} p &= (2m-2k+1)-q, && m-k+1 \leq q \leq 2m-2k, \\ &= (4m-2k)-q, && 2m-2k < q \leq 2m-k-1, \\ &= n, && 2m-k-1 < q, \end{aligned}$$

(b) for $k = m,\ m+1,\ldots,\ 2m-1,$

$$q = 4m-n-k,\ 4m-n-k+1,\ \ldots,\ 3m-k-1$$

$$\begin{aligned} p &= n, && q < 2m-k+1, \\ &= (4m-2k)-q, && 2m-k+1 \leq q \leq 4m-2k-1, \\ &= (6m-2k-1)-q, && 4m-2k-1 < q. \end{aligned}$$

The remaining elements of R_k are zero except for n odd, then $R^{(k)}_{2m-k,\,2m-k} = 1$.

For a given k, the angles $\alpha^{(k)}_{pq}$ are determined for all (p,q) such that $\alpha^{(k)}_{pq}$ eliminates the element $a^{(k)}_{pq}$; see Appendix A.

For example, in a given sweep, denoting each element in the transformation by the integer k, the patterns of the eliminated elements for matrices of orders 8 and 7 are shown below.

$$
\begin{array}{ccccccc|c}
* & 3 & 6 & 2 & 5 & 1 & 4 & 7 \\
 & * & 2 & 5 & 1 & 4 & 7 & 6 \\
 & & * & 1 & 4 & 7 & 3 & 5 \\
 & & & * & 7 & 3 & 6 & 4 \\
 & & & & * & 6 & 2 & 3 \\
 & & & & & * & 5 & 2 \\
 & & & & & & * & 1 \\
 & & & & & & & *
\end{array}
$$

7 x 7

8 x 8

Note that since $a^{(k)}_{qp}$ as well as $a^{(k)}_{pq}$ is eliminated, if one completes the lower diagonal portion of the matrix above, it is evident that any given k appears in each row and column once and only once.

For further examples of particular orthogonal transformations constructed by this elimination scheme see Appendix B .

4.0 JACOBI - An ILLIAC IV Routine

4.1 Introduction

JACOBI is an ILLIAC IV routine written in the assembly language ASK, which implements the modified Jacobi algorithm discussed in Section 3.2.

The program accepts as input a matrix A, and n, the order of the matrix, and returns as output the matrix A reduced to diagonal form, the matrix of eigenvectors corresponding to the computed eigenvalues and the number of sweeps required to achieve convergence.

A flow chart, a description of JACOBI and auxillary routines, and a short discussion of JACOBI for the potential user are now presented.

4.2 Procedure

Each sweep requires 2m-1 orthogonal transformations as described in Section 3.2. For each transformation the following sequences of events occur:

(i) The pairs (p,q) corresponding to the element $a_{pq}^{(k-1)}$ to be eliminated are determined.

(ii) The orthogonal transformation matrix R_k is constructed in order to eliminate the proper elements of A.

(iii) A_{k-1} is pre- and post-multiplied by the transformation matrix R_k to yield $A_k = R_k^t \, A_{k-1} \, R_k$ with elements $a_{pq}^{(k)} = 0$.

(iv) E_{k-1}, the eigenvector matrix, is pre-multiplied by R_k^t to yield E_k, where E_o is the identity matrix. (Note that the rows and columns of E correspond to the left and right eigenvectors respectively).

After 2m-1 such transformations have been applied, the following convergence criterion are subjected to A_k:

(a) if the sum of the squares of the off-diagonal elements is zero, convergence is attained.

(b) if the ratio of the sum of the squares of the off-diagonal elements to the sum of the squares of the diagonal elements at step k is sufficiently small (10^{-16}) in comparison to an equivalent ratio at step 1, the method has converged. (to insure numerical stability this critera is not applied for the first three sweeps).

If the convergence tests fail, a new sweep is initiated. When the method does converge, the bounds on the eigenvalues are computed using Gerschgorin discs, and this information is output for the user.

4.3 Main Flowchart

Notation: A_k - Matrix to be diagonalized at step k

E_k - Corresponding eigenvector matrix

R_k - Orthogonal transformation matrix

B - Temporary matrix

Ratio: $$\sum_{\substack{i,j \\ i \neq j}} a_{ij}^{(k)\,2} \Big/ \sum_{i=1}^{n} a_{ii}^{(k)\,2} \quad \text{where } a_{ij}^{(k)} \in A_k$$

Kconv: $$\sum_{\substack{i,j \\ i \neq j}} a_{ij}^{(1)\,2} \Big/ \sum_{i=1}^{n} a_{ii}^{(1)\,2}$$

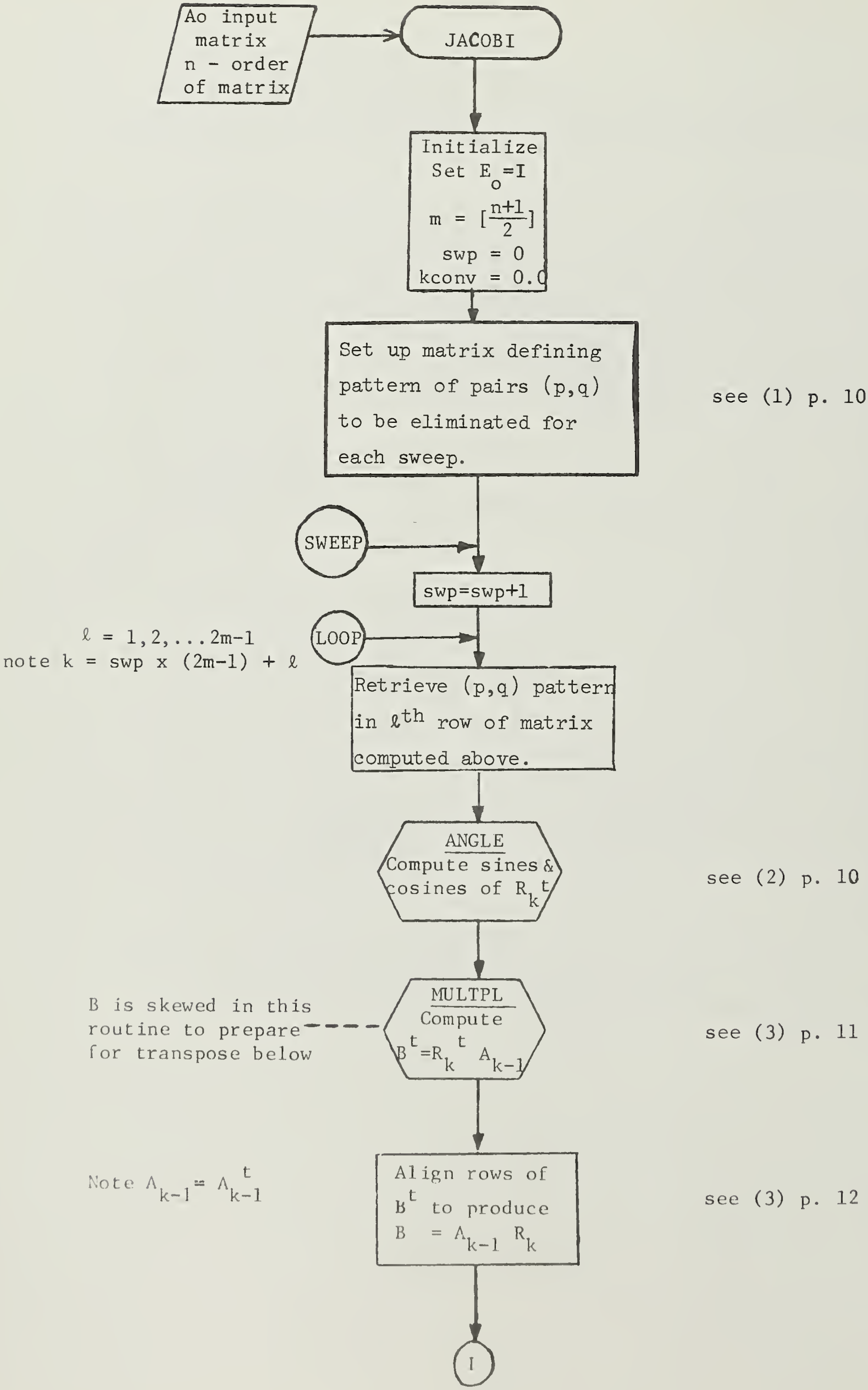
ℓ - Transformation count

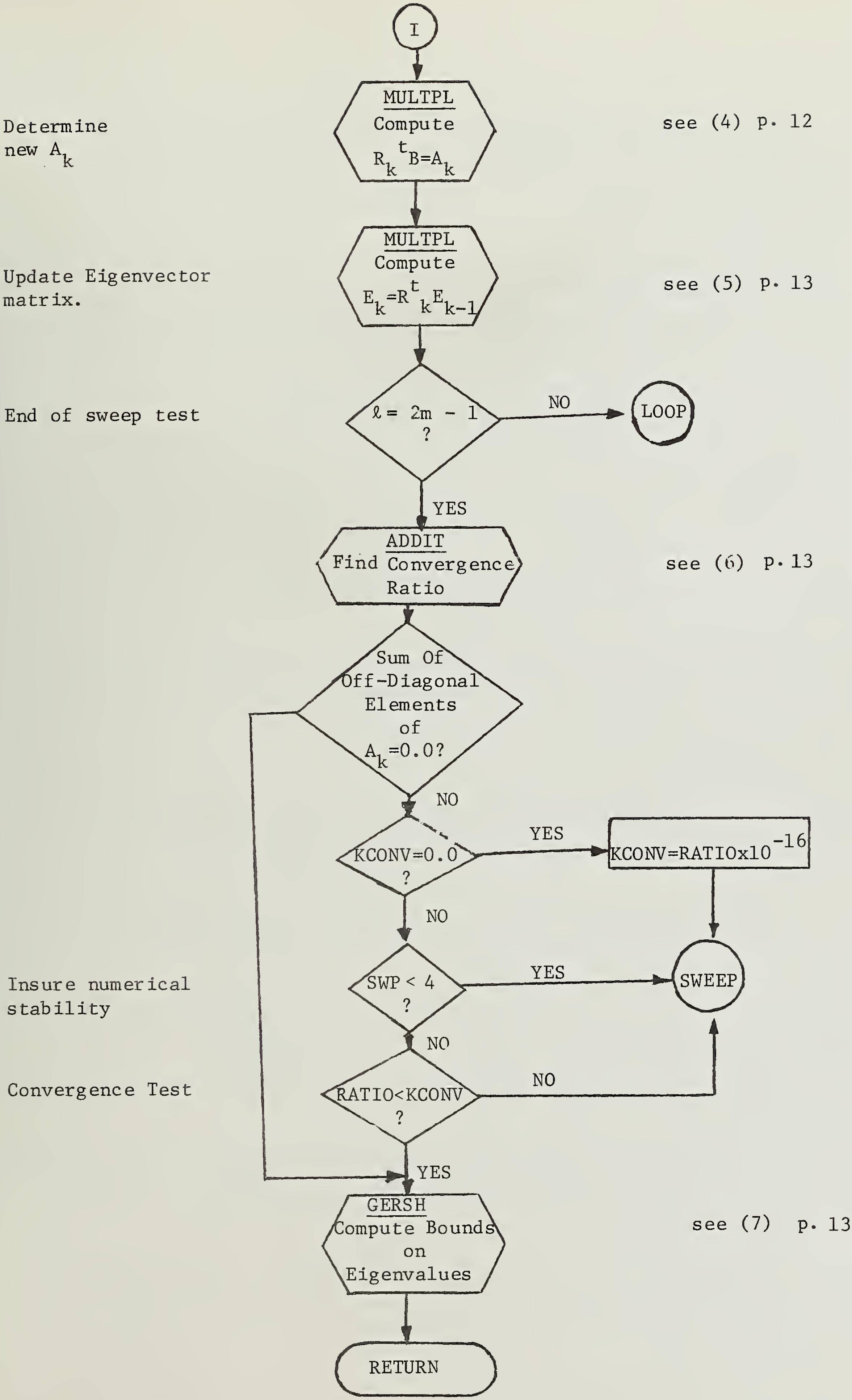
swp - Sweep count

k - swp x(2m-1) + ℓ where $m = \left[\frac{n+1}{2}\right]$

n = order of matrix

A detailed discussion of the components of JACOBI, depicted in the flowchart presented in this section may be found in Section 4.4.


Ao input matrix
n - order of matrix
JACOBI
Initialize
Set $E_o = I$
$m = [\frac{n+1}{2}]$
swp = 0
kconv = 0.0
Set up matrix defining pattern of pairs (p,q) to be eliminated for each sweep.
see (1) p. 10
SWEEP
swp=swp+1
$\ell = 1, 2, \ldots 2m-1$
note $k = swp \times (2m-1) + \ell$
LOOP
Retrieve (p,q) pattern in ℓ^{th} row of matrix computed above.
ANGLE
Compute sines & cosines of $R_k{}^t$
see (2) p. 10
B is skewed in this routine to prepare for transpose below
MULTPL
Compute
$B^t = R_k{}^t A_{k-1}$
see (3) p. 11
Note $A_{k-1} = A_{k-1}^t$
Align rows of B^t to produce
$B = A_{k-1} R_k$
see (3) p. 12
I


I
MULTPL
Compute
$R_k{}^tB=A_k$
Determine new A_k
see (4) p. 12
MULTPL
Compute
$E_k=R^t{}_kE_{k-1}$
Update Eigenvector matrix.
see (5) p. 13
End of sweep test
$\ell = 2m - 1$?
NO
LOOP
YES
ADDIT
Find Convergence Ratio
see (6) p. 13
Sum Of Off-Diagonal Elements of A_k=0.0?
NO
KCONV=0.0 ?
YES
KCONV=RATIOx10^{-16}
NO
Insure numerical stability
SWP < 4 ?
YES
SWEEP
NO
Convergence Test
RATIO<KCONV ?
NO
YES
GERSH
Compute Bounds on Eigenvalues
see (7) p. 13
RETURN

4.4 Description of Main and Auxillary Routines

After saving the necessary registers, return addresses, and addresses of calling parameters, setting up constants and counters, JACOBI constructs $E_o = I$, the identity matrix.

(1) Determination of pairs (p,q)

In order to implement this phase of the algorithm on ILLIAC IV, it is desirable to maintain compatibility with PE numbering (i.e. $0 \le p, q \le n-1$) and also alter the definition for p,q. The following definition will produce identical pairs except that now a_{ij} in Section 3.2.1 corresponds to $a_{i-1,j-1}$ as defined in this section.

Let q = PE number $\qquad q = 0,1,\ldots n-1$

$m = [\frac{n+1}{2}] \qquad n$ = dimension of matrix

For $k = 1,2,\ldots 2m-1$ let $k_o = 4m-2 \qquad k = 1,2,\ldots m-1$

$k_o = 6m-3 \qquad k = m,m+1,\ldots 2m-1$

Then $p = (k_o - 2k - q) \bmod (2m-1)$

Thus, (p,q) are defined above except for the following cases:

(a) if n even, set $p = n-1$ in $PE(n - 1 - k)$
set $p = n - 1 - k$ in $PE(n - 1)$

(b) if n odd, note that $p = q$ in one PE. This fact will be taken into consideration later on in the program.

As the pattern of pairs is constant for each sweep this calculation is only done once in the program and saved for subsequent usage.

At label SWEEP, all necessary preparations are made for another 2m - 1 transformations.

(2) ANGLE - Compute sines and cosines of the transformation matrix. Input to this routine is the matrix A_k and the pairs (p,q) determined in (1) above.

Element a_{pq} is brought to PE q. This is accomplished by a right route of (q-p) for $p < q$ or by a left route of (p-q) for $p > q$.

The sines and cosines are computed using the formulas in Appendix A and stored in two rows of PE memory (SIN , COS). If n is odd, the q^{th} PE has p = q and in this PE, cosine and sine are set to 1.0 and 0.0 respectively.

(3) <u>MULTPL</u> - Compute $B = A_{k-1}R_k$. Let n = 4, for k = 1 the ordered pairs are (0,1), (2,3). Let $R_{ij}^{(k)}$ be as defined in Section 3.2.1 with modifications to that definition as noted in (1) of this section. We wish to compute:

$$B = A_{k-1}R_k = \begin{bmatrix} a_{00}^{(k-1)} & a_{01}^{(k-1)} & a_{02}^{(k-1)} & a_{03}^{(k-1)} \\ a_{01}^{(k-1)} & a_{11}^{(k-1)} & a_{12}^{(k-1)} & a_{13}^{(k-1)} \\ a_{02}^{(k-1)} & a_{12}^{(k-1)} & a_{22}^{(k-1)} & a_{23}^{(k-1)} \\ a_{03}^{(k-1)} & a_{13}^{(k-1)} & a_{23}^{(k-1)} & a_{33}^{(k-1)} \end{bmatrix} \begin{bmatrix} R_{00} & R_{01} & 0 & 0 \\ -R_{01} & R_{11} & 0 & 0 \\ 0 & 0 & R_{22} & R_{23} \\ 0 & 0 & -R_{23} & R_{33} \end{bmatrix}$$

We note that column 1 of B = [R_{00} x col 1 of A]+[($-R_{01}$) x col 2 of A]
column 2 of B = [R_{01} x col 1 of A]+[R_{11} x col 2 of A]
etc.

Rather than working with columns it is preferable to work with rows. and without loss of generality the transformation matrix R above may now be considered as R^t.

Then $B^t = (AR)^t = R^tA^t = R^tA$ (since A is symmetric)

$\therefore B = (R^tA)^t$

$$B^t = R_k^t A_{k-1} = \begin{bmatrix} R_{00} & R_{01} & 0 & 0 \\ -R_{01} & R_{11} & 0 & 0 \\ 0 & 0 & R_{22} & R_{23} \\ 0 & 0 & -R_{23} & R_{33} \end{bmatrix} \begin{bmatrix} a_{00}^{(k-1)} & a_{01}^{(k-1)} & a_{02}^{(k-1)} & a_{03}^{(k-1)} \\ a_{01}^{(k-1)} & a_{11}^{(k-1)} & a_{12}^{(k-1)} & a_{13}^{(k-1)} \\ a_{02}^{(k-1)} & a_{12}^{(k-1)} & a_{22}^{(k-1)} & a_{23}^{(k-1)} \\ a_{03}^{(k-1)} & a_{13}^{(k-1)} & a_{23}^{(k-1)} & a_{33}^{(k-1)} \end{bmatrix}$$

In PE memory we have:

	PE 0	PE 1	PE 2	PE 3
row COS:	R_{00}	R_{11}	R_{22}	R_{33}
row SIN:	R_{01}	$-R_{01}$	R_{23}	$-R_{23}$
row PROW(p):	1	0	3	2
row PEN(q):	0	1	2	3
Row A (0):	A_{00}	A_{01}	A_{02}	A_{03}
A (1):	A_{01}	A_{11}	A_{12}	A_{13}
A (2):	A_{02}	A_{12}	A_{22}	A_{23}
A (3):	A_{03}	A_{13}	A_{23}	A_{33}

The computation of B^t is simply done as follows:

Row q of B^t = element q of row "COS" x row q of A
+ element q of row "SIN" x row p of A.

As each row of B^t is computed it is skewed in preparation for realignment to yield B. The main routine needs only to shift row i left by i (i=0,1,...n-1) to achieve the desired matrix:

$$\begin{bmatrix} b_{00} & b_{01} & b_{02} & b_{03} \\ b_{13} & b_{10} & b_{11} & b_{12} \\ b_{22} & b_{23} & b_{20} & b_{21} \\ b_{31} & b_{32} & b_{33} & b_{30} \end{bmatrix} \qquad \begin{bmatrix} b_{00} & b_{01} & b_{02} & b_{03} \\ b_{10} & b_{11} & b_{12} & b_{13} \\ b_{20} & b_{21} & b_{22} & b_{23} \\ b_{30} & b_{31} & b_{32} & b_{33} \end{bmatrix}$$

B skewed
(output from MULTPL)

B re-aligned
(in main routine)

(4) <u>Compute $R_k^t B = A_k$.</u> The routine MULTPL is employed to determine the new A_k. Naturally the skewing logic in MULTPL, described in (3) is bypassed.

(5) Compute $R_k^t E_{k-1} = E_k$. The eigenvector matrix is updated using routine MULTPL once more, and of course bypassing the skewing logic. It is preferable to pre-multiply, as a result the rows and columns of E_k will contain the left and right eigenvectors corresponding to the diagonal elements of A_k.

(6) Convergence (see Section 4.2).

(7) GERSH - Finally the bounds of the eigenvalues are determined in routine GERSH by Gerschgorin discs. The eigenvalue is the center of the disc, and the bound on eigenvalue i is the sum of the off-diagonal elements of row i in the diagonal matrix A_k.

4.5 Program Utilization

The usage of JACOBI is enabled by the call statement below:

CALL JACOBI (< the $\left\{\begin{smallmatrix}A\\ \text{diagonal}\end{smallmatrix}\right\}$ matrix >,
< temporary matrix 1 >,
< temporary matrix 2 >,
< eigenvector matrix >,
< bounds on eigenvalues >,
< order of matrix >,
< sweep count >);

All parameters are passed as addresses whose contents are described as follows:

1. < the $\left\{\begin{smallmatrix}A\\ \text{diagonal}\end{smallmatrix}\right\}$ matrix > - As input this is the original symmetric matrix to be diagonalized. If the user desires to display the original matrix or retain its contents, he should make necessary preparations before the call to JACOBI. The diagonal matrix replaces the A matrix and is output via this calling parameter.

2. < temporary matrix 1 > - a temporary matrix to enable matrix multiplication for JACOBI which is available to the user after exiting JACOBI.

3. < temporary matrix 2 > - a temporary matrix to contain the pattern describing the elements to be annihilated in a given sweep. This matrix is also available for usage upon exiting the routine.

4. < eigenvector matrix > - the matrix of eigenvectors computed by JACOBI whose rows contain the right eigenvectors and columns the left eigenvectors.

5. < bounds on eigenvalues > - a vector whose value in PE i gives the bound on λ_i.

6. < order of matrix > - the order of the matrix, an integer ≤ 64.

7. < sweep count > - the number of sweeps to achieve convergence, an integer value.

In accordance with ILLIAC IV subroutine linkage standards, the contents of ACARS 0 and 1, as well as \$D32-\$D63 are saved. The user is advised not to use \$D0-\$D31 since they will be overwritten.

PARAMETER	STORAGE REQUIRED	STORAGE MODE	RELOCATABLE	CONTENTS DESTROYED
< the $\left\{ {\text{A} \atop \text{diagonal}} \right\}$ matrix >	N* rows	Straight	Yes	Yes
< temporary matrix 1 >	N rows	-	Yes	Yes
< temporary matrix 2 >	N rows	-	Yes	Yes
< eigenvector matrix >	N rows	Straight	Yes	Yes
< bounds on eigenvalues >	1 row	Straight	Yes	Yes
< order of matrix >	1 word	-	Yes	No
< sweep count >	1 word	-	Yes	Yes

* where N = < order of matrix >

5.0 Test Results*

5.1 System of Even Order

See [2] p. 55. The matrix to be diagonalized is

$$\begin{bmatrix} 5 & 4 & 1 & 1 \\ 4 & 5 & 1 & 1 \\ 1 & 1 & 4 & 2 \\ 1 & 1 & 2 & 4 \end{bmatrix} \qquad \begin{aligned} \lambda_1 &= 10.0 \\ \lambda_2 &= 5.0 \\ \lambda_3 &= 2.0 \\ \lambda_4 &= 1.0 \end{aligned}$$

$$x_1 = \begin{bmatrix} 2 \\ 2 \\ 1 \\ 1 \end{bmatrix} \quad x_2 = \begin{bmatrix} -1 \\ -1 \\ 2 \\ 2 \end{bmatrix} \quad x_3 = \begin{bmatrix} 0 \\ 0 \\ -1 \\ 1 \end{bmatrix} \quad x_4 = \begin{bmatrix} -1 \\ 1 \\ 0 \\ 0 \end{bmatrix}$$

Output from JACOBI (note the eigenvector matrix could be scaled to yield the above result):

* Due to the slow speed of the ILLIAC IV Simulator (about 10^6 times slower than the ILLIAC IV), we tested only matrices of small order.

THE ORIGINAL MATRIX

```
##=THE ORIGINAL MATRIX###

MEMORY:
------
LOCATION:8  C(LOCATION)                     C(LOCATION +0000 01:8)     C(LOCATION +0000 02:8)     C(LOCATION +0000 03:8)
0200 00:+.5000000000000000@+0001    +.4000000000000000@+0001   +.1000000000000000@+0001   +.1000000000000000@+0001

MEMORY:
------
LOCATION:8  C(LOCATION)                     C(LOCATION +0000 01:8)     C(LOCATION +0000 02:8)     C(LOCATION +0000 03:8)
0201 00:+.4000000000000000@+0001    +.5000000000000000@+0001   +.1000000000000000@+0001   +.1000000000000000@+0001

MEMORY:
------
LOCATION:8  C(LOCATION)                     C(LOCATION +0000 01:8)     C(LOCATION +0000 02:8)     C(LOCATION +0000 03:8)
0202 00:+.1000000000000000@+0001    +.1000000000000000@+0001   +.4000000000000000@+0001   +.2000000000000000@+0001

MEMORY:
------
LOCATION:8  C(LOCATION)                     C(LOCATION +0000 01:8)     C(LOCATION +0000 02:8)     C(LOCATION +0000 03:8)
0203 00:+.1000000000000000@+0001    +.1000000000000000@+0001   +.2000000000000000@+0001   +.4000000000000000@+0001
```

THE DIAGONAL MATRIX

```
###THE DIAGONAL MATRIX###

MEMORY:

------

LOCATION:8   C(LOCATION)                        C(LOCATION +0000 01:8)          C(LOCATION +0000 02:8)          C(LOCATION +0000 03:8)
0200 00:+.99999999999998860+0001         0                              0                               0

MEMORY:

------

LOCATION:8   C(LOCATION)                        C(LOCATION +0000 01:8)          C(LOCATION +0000 02:8)          C(LOCATION +0000 03:8)
0201 00:0                               +.1000000000000014@+0001        0                               0

MEMORY:

------

LOCATION:8   C(LOCATION)                        C(LOCATION +0000 01:8)          C(LOCATION +0000 02:8)          C(LOCATION +0000 03:8)
0202 00:0                                0                              +.5000000000000000@+0001        0

MEMORY:

------

LOCATION:8   C(LOCATION)                        C(LOCATION +0000 01:8)          C(LOCATION +0000 02:8)          C(LOCATION +0000 03:8)
0203 00:0                                0                              0                               +.1999999999999986@+0001
```

THE EIGENVECTOR MATRIX

```
###THE EIGENVECTOR MATRIX

MEMORY:

------

LOCATION:8  C(LOCATION)                       C(LOCATION +0000 01:8)         C(LOCATION +0000 02:8)         C(LOCATION +0000 03:8)
0211 00:+.632455532033674 2@+0000       +.6324555320336742@+0000       +.3162277660168336@-0000       +.3162277660168336@-0000

MEMORY:

------

LOCATION:8  C(LOCATION)                       C(LOCATION +0000 01:8)         C(LOCATION +0000 02:8)         C(LOCATION +0000 03:8)
0212 00:-.7071067811865461@+0000        +.7071067811865461@+0000       0                              0

MEMORY:

------

LOCATION:8  C(LOCATION)                       C(LOCATION +0000 01:8)         C(LOCATION +0000 02:8)         C(LOCATION +0000 03:8)
0213 00:-.3162277660168336@-0000        -.3162277660168336@-0000       +.6324555320336742@+0000       +.6324555320336742@+0000

MEMORY:

------

LOCATION:8  C(LOCATION)                       C(LOCATION +0000 01:8)         C(LOCATION +0000 02:8)         C(LOCATION +0000 03:8)
0214 00:0                               0                              -.7071067811865461@+0000       +.7071067811865461@+0000
```

5.2 System of Odd Order

See [2] pp. 58-59 the matrix to be diagonalized is

$$\begin{bmatrix} 5 & 4 & 3 & 2 & 1 \\ 4 & 6 & 0 & 4 & 3 \\ 3 & 0 & 7 & 6 & 5 \\ 2 & 4 & 6 & 8 & 7 \\ 1 & 3 & 5 & 7 & 9 \end{bmatrix} \qquad \begin{aligned} \lambda_1 &= 22.40687532 \\ \lambda_2 &= 7.513724155 \\ \lambda_3 &= 4.848950120 \\ \lambda_4 &= 1.327045605 \\ \lambda_5 &= 1.096595181 \end{aligned}$$

THE ORIGINAL MATRIX

```
###THE ORIGINAL MATRIX###

MEMORY:

------

LOCATION:8  C(LOCATION)                      C(LOCATION +0000 01:8)      C(LOCATION +0000 02:8)      C(LOCATION +0000 03:8)
0200 00:+.5000000000000000@+0001   +.4000000000000000@+0001   +.3000000000000000@+0001   +.2000000000000000@+0001
0200 04:+.1000000000000000@+0001   0                          0                          0

MEMORY:

------

LOCATION:8  C(LOCATION)                      C(LOCATION +0000 01:8)      C(LOCATION +0000 02:8)      C(LOCATION +0000 03:8)
0201 00:+.4000000000000000@+0001   +.6000000000000000@+0001   0                          +.4000000000000000@+0001
0201 04:+.3000000000000000@+0001   0                          0                          0

MEMORY:

------

LOCATION:8  C(LOCATION)                      C(LOCATION +0000 01:8)      C(LOCATION +0000 02:8)      C(LOCATION +0000 03:8)
0202 00:+.3000000000000000@+0001   0                          +.7000000000000000@+0001   +.6000000000000000@+0001
0202 04:+.5000000000000000@+0001   0                          0                          0

MEMORY:

------

LOCATION:8  C(LOCATION)                      C(LOCATION +0000 01:8)      C(LOCATION +0000 02:8)      C(LOCATION +0000 03:8)
0203 00:+.2000000000000000@+0001   +.4000000000000000@+0001   +.6000000000000000@+0001   +.8000000000000000@+0001
0203 04:+.7000000000000000@+0001   0                          0                          0

MEMORY:

------

LOCATION:8  C(LOCATION)                      C(LOCATION +0000 01:8)      C(LOCATION +0000 02:8)      C(LOCATION +0000 03:8)
0204 00:+.1000000000000000@+0001   +.3000000000000000@+0001   +.5000000000000000@+0001   +.7000000000000000@+0001
0204 04:+.9000000000000000@+0001   0                          0                          0
```

THE DIAGONAL MATRIX

```
###THE DIAGONAL MATRIX###

MEMORY:

------

LOCATION:8  C(LOCATION)                        C(LOCATION +0000 01:8)          C(LOCATION +0000 02:8)          C(LOCATION +0000 03:8)
0200 00:-.1096595181658998@+0001           0                               0                               -.2345578481733705@-0016
0200 04:0                                  0                               0                               0

MEMORY:

------

LOCATION:8  C(LOCATION)                        C(LOCATION +0000 01:8)          C(LOCATION +0000 02:8)          C(LOCATION +0000 03:8)
0201 00:0                                  +.1327045599556840@+0001        -.2356991570797998@-0015        0
0201 04:0                                  0                               0                               0

MEMORY:

------

LOCATION:8  C(LOCATION)                        C(LOCATION +0000 01:8)          C(LOCATION +0000 02:8)          C(LOCATION +0000 03:8)
0202 00:0                                  -.2356991570797998@-0015        +.7513724154206159@+0001        0
0202 04:+.1404876453160047@-0022           0                               0                               0

MEMORY:

------

LOCATION:8  C(LOCATION)                        C(LOCATION +0000 01:8)          C(LOCATION +0000 02:8)          C(LOCATION +0000 03:8)
0203 00:-.2345578481733705@-0016           0                               0                               +.4848950120316658@+0001
0203 04:0                                  0                               0                               0

MEMORY:

------

LOCATION:8  C(LOCATION)                        C(LOCATION +0000 01:8)          C(LOCATION +0000 02:8)          C(LOCATION +0000 03:8)
0204 00:0                                  0                               +.1404876453160047@-0022        0
0204 04:+.2240687530758328@+0002           0                               0                               0
```

BOUNDS ON EIGENVALUES

```
### PL I CONTAINS BOUND ON EIGENVAL, I ###

MEMORY:

------

LOCATION:8  C(LOCATION)                        C(LOCATION +0000 01:8)          C(LOCATION +0000 02:8)          C(LOCATION +0000 03:8)
0227 00:+.2345578481733705@-0016           +.2356991570797998@-0015        +.2356991711285638@-0015        +.2345578481733705@-0016
0227 04:+.1404876453160047@-0022           0                               0                               0
```

THE EIGENVECTOR MATRIX

```
###THE EIGENVECTOR MATRIX

MEMORY:

------

LOCATION:8  C(LOCATION)                     C(LOCATION +0000 01:8)       C(LOCATION +0000 02:8)       C(LOCATION +0000 03:8)
0212 00:+.4693580751718827@-0000    -.5422121959852078@+0000     -.5444524035263782@+0000     +.4258656563603971@-0000
0212 04:+.8898850950013593@-0001    0                            0                            0

MEMORY:

------

LOCATION:8  C(LOCATION)                     C(LOCATION +0000 01:8)       C(LOCATION +0000 02:8)       C(LOCATION +0000 03:8)
0213 00:+.3410130371516615@-0000    -.1164346155612828@+0000     -.1959066716037894@-0001     -.6820430386473433@+0000
0213 04:+.6360712129822161@+0000    0                            0                            0

MEMORY:

------

LOCATION:8  C(LOCATION)                     C(LOCATION +0000 01:8)       C(LOCATION +0000 02:8)       C(LOCATION +0000 03:8)
0214 00:-.5509619554095337@+0000    -.7094403390575579@+0000     +.3401791338957239@-0000     +.8341095366055651@-0001
0214 04:+.2654356771059980@-0000    0                            0                            0

MEMORY:

------

LOCATION:8  C(LOCATION)                     C(LOCATION +0000 01:8)       C(LOCATION +0000 02:8)       C(LOCATION +0000 03:8)
0215 00:+.5471727959844159@+0000    -.3125699199737362@-0000     +.6181120764313803@+0000     -.1156065934015595@+0000
0215 04:-.4554937463830342@-0000    0                            0                            0

MEMORY:

------

LOCATION:8  C(LOCATION)                     C(LOCATION +0000 01:8)       C(LOCATION +0000 02:8)       C(LOCATION +0000 03:8)
0216 00:+.2458779184757240@+0000    +.3023960473727580@-0000     +.4532145241990548@-0000     +.5771771924331297@+0000
0216 04:+.5563845462713388@+0000    0                            0                            0
```

5.3 Comparison with Existing Jacobi Algorithm

W. Bernhard's ILLIAC IV routine EIGEN [1] is essentially the algorithm discussed briefly in Section 3.1. A comparison run was performed to insure the two algorithms produced compatible results. See [1] pp. 127-129.

Output from JACOBI:

THE ORIGINAL MATRIX

```
********** DISPLAY #        1     ICR=  002443 1:16  TIME=09:27:44:00     ELAPSED PROCESSOR TIME=00:00:05:11*************************

MEMORY:
------
LOCATION:8  C(LOCATION)                   C(LOCATION +0000 01:8)          C(LOCATION +0000 02:8)          C(LOCATION +0000 03:8)
0200 00:+.3999997070700005E+0001       -.3058792135850006E-0003         -.2407922355729994E-0002         +.3289387772360009E-0004

********** DISPLAY #        2     ICR=  002443 1:16  TIME=09:27:44:58     ELAPSED PROCESSOR TIME=00:00:05:24*************************

MEMORY:
------
LOCATION:8  C(LOCATION)                   C(LOCATION +0000 01:8)          C(LOCATION +0000 02:8)          C(LOCATION +0000 03:8)
0201 00:-.3058792135850006E-0003       +.1000000040880003E+0001         -.3945657481569992E-0005         -.5945699681600002E-0004

********** DISPLAY #        3     ICR=  002443 1:16  TIME=09:27:45:06     ELAPSED PROCESSOR TIME=00:00:05:31*************************

MEMORY:
------
LOCATION:8  C(LOCATION)                   C(LOCATION +0000 01:8)          C(LOCATION +0000 02:8)          C(LOCATION +0000 03:8)
0202 00:-.2407922355729994E-0002       -.3945657481569992E-0005         +.2000002920669997E+0001         -.1721223952379997E-0003

********** DISPLAY #        4     ICR=  002443 1:16  TIME=09:27:45:15     ELAPSED PROCESSOR TIME=00:00:05:38*************************

MEMORY:
------
LOCATION:8  C(LOCATION)                   C(LOCATION +0000 01:8)          C(LOCATION +0000 02:8)          C(LOCATION +0000 03:8)
0203 00:+.3289387772360009E-0004       -.5945699681600002E-0004         -.1721223952379997E-0003         +.2999999730400005E+0001
```

THE DIAGONAL MATRIX

```
LOCATION:8  C(LOCATION)                    C(LOCATION +0000 01:8)        C(LOCATION +0000 02:8)        C(LOCATION +0000 03:8)
0200 00:+.4000000000203149 6@+0001         +.6048634568778123@-0008      +.1314960543097658@-0011      0

MEMORY:
------
LOCATION:8  C(LOCATION)                    C(LOCATION +0000 01:8)        C(LOCATION +0000 02:8)        C(LOCATION +0000 03:8)
0201 00:+.6048634568778097@-0001           +.1000000000797659@+0001      0                             -.3247599414537152@-0013

MEMORY:
------
LOCATION:8  C(LOCATION)                    C(LOCATION +0000 01:8)        C(LOCATION +0000 02:8)        C(LOCATION +0000 03:8)
0202 00:+.1314960543097658@-0011           0                             +.1999999992247034@+0001      +.4023461519379488@-0007

MEMORY:
------
LOCATION:8  C(LOCATION)                    C(LOCATION +0000 01:8)        C(LOCATION +0000 02:8)        C(LOCATION +0000 03:8)
0203 00:0                                  -.3247599414537071@-0013      +.402346151937969@-0007       +.3000000133237066@+0001
```

THE EIGENVECTOR MATRIX

```
MEMORY.
------
LOCATION:8  C(LOCATION)                     C(LOCATION +0000 01:8)         C(LOCATION +0000 02:8)         C(LOCATION +0000 03:8)
0213 00:+.99999926949120327+0000            -.10196077224531798-0003       -.12039647070288618-0002       +.33107129387257520-0004

MEMORY:
------
LOCATION:8  C(LOCATION)                     C(LOCATION +0000 01:8)         C(LOCATION +0000 02:8)         C(LOCATION +0000 03:8)
0214 00:+.10196[illegible]                  +.[illegible]350933[illegible]+0000  +.41960564977452966-0005  +.29727[illegible]01028399810-0004

MEMORY.
------
LOCATION:8  C(LOCATION)                     C(LOCATION +0000 01:8)         C(LOCATION +0000 02:8)         C(LOCATION +0000 03:8)

0215 00:+.12039585567[illegible]-0002       -.[illegible]3239316725993188-0005  +.99999926041919108+0000  +.17212[illegible]017936378-0003

MEMORY:
------
LOCATION:8  C(LOCATION)                     C(LOCATION +0000 01:8)         C(LOCATION +0000 02:8)         C(LOCATION +0000 03:8)
0216 00:-.3331[illegible]-000[illegible]    -.2972307145516297[illegible]-0004  -.17208280052300958-0003  +.99999[illegible]197007308+0000

***************************************************************************************************************
```

References

1. Bernhard, W. "ILLIAC IV Codes for Jacobi and Jacobi-Like Algorithms." CAC Document No. 19. Center for Advanced Computation, University of Illinois at Urbana-Champaign, Urbana, Illinois, November 1971.

2. Gregory, R. T., and D. L. Karvey. A Collection of Matrices for Testing Computational Algorithms. Wiley-Interscience, New York, 1969.

3. Sameh, A. "On Jacobi and Jacobi-Like Algorithms for a Parallel Computer," Mathematics of Computation. Vol. 25, No. 115, pp. 579-590, July 1971.

4. Sameh, A., and L. Han. "Eigenvalue Problems." ILLIAC IV Document No. 127. Department of Computer Science, University of Illinois at Urbana-Champaign, Urbana, Illinois, 1968.

5. Wilkinson, J. H. The Algebraic Eigenvalue Problem, Clarendon Press, Oxford, 1965.

APPENDIX A   Determination of the Rotation Angle

The orthogonal matrix $R(p,q,\alpha_{pq}^{(k)})$, differs from the identity matrix by a 2 x 2 diagonal submatrix whose elements are

(A.1) $$R_{pp} = R_{qq} = \cos\alpha_{pq}^{(k)} \qquad R_{pq} = -R_{qp} = \sin\alpha_{pq}^{(k)}$$

where $p < q$. In order to eliminate the off-diagonal element $a_{pq}^{(k)}$, the angle α_{pq} is chosen such that

(A.2) $$\tan 2\alpha_{pq}^{(k)} = \frac{2a_{pq}^{(k)}}{a_{pp}^{(k)} - a_{qq}^{(k)}}$$

in which $\alpha_{pq}^{(k)}$ is restricted by $|\alpha_{pq}^{(k)}| \leqq /4$. Let

$$t_k = |2a_{pq}^{(k)}| , \quad x_k = |a_{pp}^{(k)} - a_{qq}^{(k)}| , \quad y_k = (t_k^2 + x_k^2)^{1/2} ;$$

then

(A.3) $$\cos^2\alpha_{pq}^{(k)} = \frac{1}{2}\left(1 + \frac{x_k}{y_k}\right); \quad \sin^2\alpha_{pq}^{(k)} = \frac{1}{2}\left(1 - \frac{x_k}{y_k}\right).$$

Since $|\alpha_{pq}^{(k)}| \leqq /4$, then $\cos\alpha_{pq}^{(k)}$ will always be taken positive and $\sin\alpha_{pq}^{(k)}$ will be of the same sign as $[2a_{pq}^{(k)}/(a_{pp}^{(k)} - a_{qq}^{(k)})]$.

Examples of Elimination Scheme [3]

Let $n = 8$ and $k = 2$, then the pairs (p,q) are given by $\{(2,3); (1,4); (7,5); (8,6)\}$ and R_2 of the form

$$\begin{bmatrix}
R_{11}^{(2)} & & & R_{14}^{(2)} & & & & \\
& R_{22}^{(2)} & R_{23}^{(2)} & & & & & \\
& -R_{23}^{(2)} & R_{33}^{(2)} & & & & & \\
-R_{14}^{(2)} & & & R_{44}^{(2)} & & & & \\
& & & & R_{55}^{(2)} & & R_{57}^{(2)} & \\
& & & & & R_{66}^{(2)} & & R_{68}^{(2)} \\
& & & & -R_{57}^{(2)} & & R_{77}^{(2)} & \\
& & & & & -R_{68}^{(2)} & & R_{88}^{(2)}
\end{bmatrix}$$

while for $k = 7$ the pairs (p,q) are $\{(8,1); (7,2); (6,3); (5,4)\}$ and R_7 is the form

$$\begin{bmatrix}
R_{11}^{(7)} & & & & & & & R_{18}^{(7)} \\
& R_{22}^{(7)} & & & & & R_{27}^{(7)} & \\
& & R_{33}^{(7)} & & & R_{36}^{(7)} & & \\
& & & R_{44}^{(7)} & R_{45}^{(7)} & & & \\
& & & -R_{45}^{(7)} & R_{55}^{(7)} & & & \\
& & -R_{36}^{(7)} & & & R_{66}^{(7)} & & \\
& -R_{27}^{(7)} & & & & & R_{77}^{(7)} & \\
-R_{18}^{(7)} & & & & & & & R_{88}^{(7)}
\end{bmatrix}$$

If the order of the matrix is odd, say $n = 7$, then for $k = 3$ the pairs (p,q) are given by $\{(1,2);\ (7,3);\ (6,4)\}$ and R_3 is of the form

$$
\begin{bmatrix}
R_{11}^{(3)} & R_{12}^{(3)} & & & & & \\
-R_{12}^{(3)} & R_{22}^{(3)} & & & & & \\
& & R_{33}^{(3)} & \cdots & \cdots & \cdots & R_{37}^{(3)} \\
& & \vdots & R_{44}^{(3)} & \cdots & R_{46}^{(3)} & \vdots \\
& & \vdots & \vdots & 1 & \vdots & \vdots \\
& & \vdots & -R_{46}^{(3)} & \cdots & R_{66}^{(3)} & \vdots \\
& & -R_{37}^{(3)} & \cdots & \cdots & \cdots & R_{77}^{(3)}
\end{bmatrix}
$$

APPENDIX C A Typical Calling Program

```
?USER=CACEIGEN
?COMPILE MCD/ASKO/JACOBIDRIVER WITH ASK LIBRARY
?BCL CARD
       BEGIN
       FILL      128;
DEFINE PRINTMTX=
       CLC(1);
       CADD(1)  $C3;
       CADD(1)  $D41;
       CSHL(1)  24;
       CADD(1)  $C3;
       CCB(1)   15;
       CLC(0);
       CADD(0)  $D41;
       CSHL(0)  24;
       CCB(0)   15;
       DISPLAYR $C1,=16;
       LIT(2)   =64;
       CADD(1)  $C2;
       CROTR(1) 24;
       CADD(1)  $C2;
       CROTL(1) 24;
       TXEFM(0) ,=9;##;
% ################END DEFINE####################
       DEFINE NMAX=16##;
MATA:  BLK      NMAX;
MATB:  BLK      NMAX;
MATP:  BLK      NMAX;
EIGV:  BLK      NMAX;
GERSH: BLK      1;
N:     WDS      1;
SWP:   WDS      1;
       DATA     0,0,0;
MESS0: DATA "###THE ORIGINAL MATRIX###";
MESS1: DATA "###THE DIAGONAL MATRIX###";
MESS2: DATA "###THE EIGENVECTOR MATRIX";
MESS3: DATA "###NUMBER OF SWEEPS REQUIRED###";
MESS4: DATA "### PE I CONTAINS BOUND ON EVAL(I) ###";
MESS5: WDS 0;
START: FILL;
%
       SETE E.OR.-E; SETE1 E.AND.E;
       LIT(0)    1,N,N;  % READ DIMENSION OF
       INPUT     $C0,1;  % SYSTEM FROM INPUT
%
       LIT(0)    =1;
       STL(0)    $D40;     % FIXED PT 1 FOR PRNTMTX DEF
       SLIT(0)   =N;
       LOAD(0)   $C0;
       CSUB(0)   $D40;
       STL(0)    $D41;     % N-1 FOR PRNTMTX DEF
%
       LIT(0)    1,MATA,MATA;
       CROTR(0) 24;
       CADD(0)  $D41;    % SET UP ACAR 0 FOR INPUT INSTRN
       CROTL(0) 24;      % USED TO READ MATA
%
```

```
       LIT(1)    0,1,0;   % ACAR 1 LOOP INDEX
       CADD(1)   $D41;    % FROM 0 TO N-1
       CROTL(1) 24;
       CLRA;              % CLEAR RGA FOR ZERO-FILLING MTX
%
RDINPT:CLC(2);            % ACAR2 USED FOR PE ROW ADDR
       CADD(2)   $C0;     % CU ADDR FROM ACARO
       CSHR(2)   6;       % PE ADDR TO $C2
       STA       0(2);    % ZERO-FILL ROW OF MATRIX
       INPUT     $C0,1;   % READ MATRIX FROM INPUT
       CROTR(0) 24;       % BUMP ACARO
       LIT(2)    =64;     % TO PREPARE
       CADD(0)   $C2;     % FOR NEXT
       CROTL(0) 24;       % ROW OF MATRIX
       CADD(0)   $C2;     % INPUT
       TXLTM(1) ,RDINPT;
%
       LIT(0)    1,MESS1-1,MESS0;
       DISPLAY   $C0,32;
       CLC(3);
       SLIT(3)   =MATA;    PRINTMTX;  % THE ORIGINAL MTX
% *******
       CALL JACOBI(MATA,MATB,MATP,EIGV,GERSH,N,SWP);
% *******
       LIT(0)    1,MESS2-1,MESS1;
       DISPLAY   $C0,32;
       SLIT(3)   =MATA; PRINTMTX;  % THE EIGENVALUES
       LIT(0)    1,MESS5-1,MESS4;
       DISPLAY   $C0,32;
       LIT(0)    1,GERSH,GERSH;
       CROTR(0) 24;
       CADD(0)   $D41;
       CROTL(0) 24;
       DISPLAYR $C0,16; % BOUNDS ON EIGENVALS
       LIT(0)    1,MESS3-1,MESS2;
       DISPLAY   $C0,32;
       SLIT(3)   =EIGV; PRINTMTX;  % THE EIGENVECTORS
       LIT(0)    1,MESS4-1,MESS3;
       DISPLAY   $C0,32;
       LIT(0)    1,MESS0-1,SWP;    % ITERATION COUNT
       DISPLAY   $C0,16;
       HALT;
       END       START.
?END
```

APPENDIX D JACOBI Listing (ASK)

```
?USER=CACEIGEN
?COMPILE MCD/ASKO/NEW.JACOBI WITH ASK LIBRARY
?DATA
        BEGIN
        FILL       128;
.ZERO: EQU        $D0;
.ONE  : EQU       $D1;
.TWO  : EQU       $D2;
.SIXT4:EQU        $D3;
.N     : EQU      $D5;         % DIMENSION OF SYSTEM
.N1    : EQU      $D6;         % N MINUS 1
.M     : EQU      $D7;         % [(N+1)/2]
.M2    : EQU      $D8;         % 2*M
.M2M1 : EQU       $D9;         % 2*M-1
.M4M2: EQU        $D10;
.SVLOOP:EQU       $D11;        % LOOP FROM 0 TO N-1
%***COUNTERS, INDICATORS AND LIMITS
.AMT:  EQU        $D12;        % ROUTING DISTANCE
.BOUND:EQU        $D13;        % INDICATOR TO ENABLE CONVERGENCE TEST
.EVNODD:EQU       $D14;        % EVEN/ODD INDICATOR FOR N
.ICNT: EQU        $D15;        % ITERATION COUNTER
.KCONV:EQU        $D16;        % CONVERGENCE FACTOR
.RATIO:EQU        $D17;        % RATIO OF SUM OF SQRS OF OFF-DIAG TO SUM OF
                               % SQUARES OF DIAG ELEMENTS
.ROUT: EQU        $D18;        % 64-N
%***SWITCHES
.ADSWT:EQU        $D19;        % =0 IF OFF-DIAG SUM, =1 IF DIAG SUM
.RSWTCH:EQU       $D20;        % ROUTING SWITCH (0=RT,1=LFT)
%***SAVE PATTERNS
.PEMODE:EQU       $D21;        % ENABLING PATTERN FOR PE-S 0 TO N-1
.SAVI: EQU        $D22;
.SAVEI:EQU        $D23;
.SVMOD:EQU        $D24;
%***SAVE REGISTERS OR ADDRESSES
.ADR1: EQU        $D25;        % ADDR OF MTX TO BE MULTPLIED
.ADR2: EQU        $D26;        % ADDR OF MTX CONTAINING RESULT OF MULT
.INDEX:EQU        $D27;        % INDEX USED IN (P,Q) SETUP
.INLOOP:EQU       $D28;
.RETUR:EQU        $D29;        % RETURN ADDR TO CALLING PROG
.SAVE0:EQU        $D30;        % SAVE REGISTERS
.SAVE1:EQU        $D31;
.SAV3: EQU        $D32;
.SAVE3:EQU        $D33;
.ADRA: EQU        $D34;        % ADDRESS OF THE INPUT MTX
.ADRB: EQU        $D35;        % ADDR OF MTX CONTAINING TEMP RESULTS
.ADRP: EQU        $D36;        % ADDR OF MTX CONTAINING PAIRS (P,Q)
.ADRE: EQU        $D37;        % ADDR OF EIGENVECTOR MTX
.ADRG: EQU        $D38;        % ADDR OF ROW TO CONTAIN BOUNDS ON EVAL
.ADRI: EQU        $D39;        % ADDR OF SWEEP COUNT FOR OUTPUT
% **** NOTE- PROG CURRENTLY EXPECTS COSA,SINA, & PROW TO BE STORED
%            IN ORDER
COSA:   BLK        1;          % COSINES OF TRANSFORM MTX
SINA:   BLK        1;          % SINES   OF TRANSFORM MTX
PROW:   BLK        1;          % ROW INDEX OF ELEMENT TO BE ANNIHILATED
TEMP1:  BLK        1;          % TEMP STORAGE
TEMP2:  BLK        1;
TEMPP:  BLK        1;
```

```
EN:   DATA 0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,
           21,22,23,24,25,26,27,28,29,30,31,32,33,34,35,36,37,38,39,40,
           41,42,43,44,45,46,47,48,49,50,51,52,53,54,55,56,57,58,59,
           60,61,62,63;
UMMY: WDS 0;   FILL 16;
DBSAV:WDS        10;        % SAVE AREA FOR $D32-$D39 AND $C0-$C1

NGLE: FILL;

 ROUTINE ANGLE COMPUTES THE SINES AND COSINES OF THE TRANSFRORMATION M
 THE COSINES AND SINES ARE STORED IN TWO ROWS OF PE MEM--COSA * SINA
 AND ARE DETERMINED BY THE VALUES PEN (Q) AND PROW (P) IN EACH PE
 ,I.E, A(P,Q), A(&,&), AND A(Q,Q) ARE DETERMINED BY P,Q.

      STL(3)     .SAVE3;    % RETURN ADDR
      LDX        PEN;
      LDL(0)     .ADRA;
      LDR        *0(0);     % MAIN DIAGONAL TO RGR
      LDA        PEN;
      SBM        PROW;
      CLC(2);
      TAG        $C2;
      SETC(1)    I;         % SET I BITS WHERE P<Q
      STL(1)     .SAVEI;
      SAP;                  % RGA = ABS(Q-P)
      IME        $C2;
      SETC(3)    I;         % IF N IS EVEN .SAVI=0
      STL(3)     .SAVI;     % IF N IS ODD .SAVI NOTES PE WHERE P=Q
      SKIP       .BUMPX2;

 BRING ELEMENT A(P,P) TO PE Q

NG1:  IME        $C2;
      SETC(3)    I;         % SEE IF RGA=$C2
      ZERT(3)    .BUMPX2;
      LDEE1      $C3;       % PE-S ON WHERE RGA=$C2
      LDL(1)     .SAVEI;
      CAND(1)    $C3;
      LDEE1      $C1;       % TURN ON PE-S WHERE RGA=$C2 AND P<Q
      RTL        0(2);      % P>Q  DO RIGHT ROUTE OF $C2 TO GET A(P,P)
      STR        TEMPP;
      LDL(1)     .SAVEI;
      COMPC(1);
      CAND(1)    $C3;
      LDEE1      $C1;       % TURN ON PE-S WHERE RGA=$C2 AND P>Q
      LDL(3)     .SIXT4;
      CSUB(3)    $C2;       % BRING RGR BACK INTO POSITION AND
      CSUB(3)    $C2;       % DO LEFT ROUTE OF $C2 (RIGHT 64-$C2)
      RTL        0(3);      % TO GET A(P,P)
      STR        TEMPP;
      LDB        *0(0);     % BRING BACK MAIN DIAGONAL
      LDL(3)     .PEMODE;
      LDEE1      $C3;       % TURN FIRST N-1 PE-S ON
UMPX2:ALIT(2)    =1;        % LOOP FROM 1 THRU N-1
      LESSTA(2) $D5,ANG1;

 COMPUTE SINES AND COSINES
```

```
%
        LIT(1)    =2.0;
        LDX       PROW;
        LDA       *0(0);
        MLRN      $C1;
        STA       TEMP1;    % TEMP1=2*A(P,Q)
        LDX       PEN;
        LDA       *0(0);
        SBRN      TEMPP;
        STA       TEMP2;    % TEMP2=A(Q,Q)-A(P,P)
        EOR       TEMP1;
        CLC(1);
        IAL       $C1;
        SETC(3)   I;        % SET I BITS WHERE TEMP1 & 2 DIFFER IN SIGN
        LDA       TEMP2;
        JLZ;                % CHECK IF ANY TEMP2=0.0
        SETC(1)   J;        % J BITS ON WHERE TEMP2=0.0
        ZERT(1)   ,NOZERO;
        LDL(0)    ,SAVET;   % SET BITS IN $C0 WHERE P<Q
        CAND(1)   $C0;      % P<Q AND TEMP2=0.0
        CEXOR(3)  $C1;      % CHANGE MODE STATUS OF THESE PE-S
        OR        TEMP1;    % SOME TEMP2=0 SEE IF CORRESPONDING
        ILZ;                % TEMP1=0 AND SET I BITS
        SETC(1)   I;        % BRING TO ACAR 1
        ZERT(1)   ,NOZERO;  % IF NO TEMP1 & 2 BOTH = 0 JUMP
        LDL(2)    .SAVI;    % NOTE THESE PE-S IN PATTERN TO SET
        COR(2)    $C1;      % COS & SIN =1.0 AND 0.0 RESPECT.
        STL(2)    .SAVI;    % WILL BE USED SHORTLY
        SETE      =I.AND.E; % TURN OFF PE=S WHERE TEMP1=TEMP2=0
        SETE1     E.AND.E;  % TO AVOIDE ZERO DIVIDE
NOZERO; STL(3)    .SVMOD;   % SAVE PATTERN
        LDA       TEMP2;
        MLRN      $A;       % TEMP2*TEMP2
        LDS       $A;
        LDA       TEMP1;
        MLRN      $A;       % TEMP1*TEMP1
        ADRN      $S;
        CALL SQRT64();      % SQRT(TEMP1**2+TEMP2**2)
        LDS       $A;
        LDA       TEMP2;
        SAP;
        DVRN      $S;
        LIT(1)    =1.0;
        IAG       $C1;
        SETC(2)   I;
        ZERT(2)   ,SVTEMP;
        LDEE1     $C2;      % INSURE ABS(COS) & ABS(SIN) LEQ 1.0
        LDA       $C1;
        LDL(2)    .PEMODE;
        LDEE1     $C2;
SVTEMP; STA       TEMPP;    % SAVE TEMP2/(ABOVE ROOT)
        ADRN      $C1;
        LIT(0)    =0.5;
        MLRN      $C0;
        CALL SQRT64();      % COSINE
        STA       COSA;
        LDA       $C1;
```

```
      SBRN      TEMPP;
      MLRN      $C0;
      CALL SQRT64();     % SINF
      STA       SINA;
      LDL(3)    .SVMOD;
      LDEE1     $C3;
      CHSA;
      STA       SINA;
      CLC(2);
      LDL(3)    .SAVI;
      LDEE1     $C3;
      LDA       $C1;
      STA       COSA;    % COS=1.0
      LDA       $C2;
      STA       SINA;    % SIN=0.0
RESET: LDL(3)   .PEMODE;
      LDEE1     $C3;
      LDL(3)    .SAVE3;
      EXCHL(3)  $ICR;
*****************************************************
*
* #########MULTIPLY ROUTINE ##############
*
*  MULTIPLY TRANSFORMATION MATRIX BY MATRIX IN ADR1--RESULT IN MATRIX IN
*  MULTIPLY IS DONE AS FOLLOWS--
*      ELEMENT :PEN: OF ROW COS * ROW :PEN: OF MTX IN ADR1
*    + ELEMENT :PEN: OF ROW SIN * ROW :PROW: OF MTX IN ADR1
*  RESULT IN ROW :PEN: OF MTX IN ADR2
*
*                    CALL 1      CALL 2      CALL3
*  MATX IN ADR1      MATA        MATB        EIGV
*  MATX IN ADR2      MATB        MATA        MATB
*
*
MULTPL:FILL;
      STL(3)    .SAVE3;
      LDL(0)    .SVLOOP; % LOOP FROM 0 TO N-1
      LDX       PEN;
MULT:  SLIT(1)  =COSA;
      CADD(1)   $C0;
      LOAD(1)   $C2;     % COS (P,Q)
      LDL(3)    .ADR1;
      CADD(3)   $C0;
      LDA       0(3);    % LOAD ROW (PEN) OF MATX IN ADR1
      MLRN      $C2;
      LDS       $A;
      CADD(1)   .SIXT4;
      LOAD(1)   $C2; % SIN(P,Q)
      CADD(1)   .SIXT4;
      LOAD(1)   $C3;     % =VALUE OF PROW IN PE 0
      LDL(1)    .ADR1;
      CADD(1)   $C3;
      LDA       0(1);    % LOAD ROW (PROW) OF MTRX IN ADR1
      MLRN      $C2;
      ADRN      $S;
*  IF THIS IS POST MULTIPLY (CALL1) - SKEW MATRIXB
      LDL(1)    .ADR1;
```

```
        LDL(2)    .ADR2;
        EQLXF(1)  $D34,MULT1;
        CLC(3);
        EQLXT(0)  $C3,MULT0;% NO NEED TO SKEW FIRST ROW
        STL(0)    .AMT;     % SKEW MATX IN ADR2 TO PEPARE FOR TRANSPOSE
        SLIT(3)   =ROUTE;
        EXCHL(3)  $ICR;
MULT0:  STA       *0(2);    % STORE RGA SKEWED IN MATX IN ADR2
        CLRA;
        LDA       $X;
        LDL(3)    $D1;
        STL(3)    .AMT;
        CLC(3);
        SLIT(3)   =ROUTE;   % ROUTE PE INDICIES 1 RIGHT
        EXCHL(3)  $ICR;
        SWAP;               % NO DIRECT PATH FOR LDX $A
        LDX       $B;
        SKIP      .CKAC0;
MULT1:  CADD(2)   $C0;
        STA       0(2); % STORE RGA IN MTX IN ADR2
CKAC0:  TXLTM(0)  .MULT;
        LDL(3)    .SAVE3;
        EXCHL(3)  $ICR;
%**********************************************************
%
% ###################ROUTE ROUTINE FOR N=64######################
%  $A HAS ELEMENTS TO BE ROUTED, .AMT CONTAINS ROUTING DISTANCE
%
ROUTE:  FILL;
        STL(0)    .SAVE0;
        STL(1)    .SAVE1;
        STL(3)    .SAV3;
        LDL(0)    .AMT;
        RTL       $A,0(0);
        LDA       $R;
        LDS       PEN;
%  SEE IF LEFT($D31=1) OR RIGHT ($D31=0) ROUTE
        CLC(1);
        EQLXT(1)  $D20,RIGHT;
        LDL(3)    .N;
        CADD(0)   .N1;
        CSUB(0)   .SIXT4;
        ISG       $C0;
        SKIP      .SETIT;
RIGHT:  LDL(3)    .ROUT; % 64-N
        ISL       $C0;
SETIT:  SETE      I.AND.E;
        SETE1     E.AND.E;
        CLRA;
        RTL       0(3);
        LDA       $R;
        LDL(3)    .PEMODE;
        LDEE1     $C3;
        LDL(0)    .SAVE0;
        LDL(1)    .SAVE1;
        LDL(3)    .SAV3;
        EXCHL(3)  $ICR;
```

```
%
%*************************************
ADDIT: FILL;
%  ADDIT CALCULATES THE SUMM OF THE OFF-DIAGONALS SQUARE AND
%  DIVIDES IT BY THE SUM OF THE DIAGONALS SQUARED
%
       STL(3)     .SAVE3;
       CLC(1);
       STL(1)     .ADSWT;
       LDL(0)     .SVLOOP;
       SETE       E.OR.-E;
       SETE1      E.AND.E;
       CLRA;
       LDS        $A;
       LDL(3)     .PEMODE;
       LDEE1      $C3;
ADI:   LDL(2)     .ADRA;
       CADD(2)    $C0;
       LDA        0(2);
       MLRN       $A;
       ADRN       $S;
       LDS        $A;
       TXEFM(0)   .ADI;      % END ROW-SUM
ADI0:  SETE E.OR.-E;
       SETE1      E.AND.E;
       LDL(0)     .ONE;
ADI1:  LDS        $A;        % SUM UP THE ELEMENTS IN EACH PE
       RTL        $S.0(0);   % USED FOR BOTH OFF-DIAGONALS AND DIAGONALS
       LDS        $R;
       ADRN       $S;
       CADD(0)    $C0;
       LDL(1)     .SIXT4;
       EQLXF(0)   $C1.ADI1;% END LOG-SUM
       LDL(1)     .ADSWT;
       EQLXT(1)   $D1.ADI2;
       STA        TEMPP;
       CLRA;
       LDS        $A;
       LDL(1)     .PEMODE;   % PICK UP THE A[I.I]-S
       LDEE1      $C1;
       LDX        PEN;
       LDL(2)     .ADRA;
       LDA        *0(2);
       MLRN       $A;
       LDL(1)     .ONE;
       STL(1)     .ADSWT;
       SKIP       .ADI0;
ADI2:  LDS        $A;
       LDA        TEMPP;
       SBRN       $S;        % SUBTRACT THE A[I.I]
       CLC(0);
       IME        $C0;
       SETC(0)    I;
       ONEST(0)   .ADI3;
       DVRN       $S;        % CONVERGENCE FACTOR
       LDL(0)     $A;
       STL(0)     .RATIO;
```

```
ADI3:  LDL(3)    .SAVE3;
       EXCHL(3) $ICR;
%*********
GERSH: FILL;              % GERSH FINDS A BOUND ON THE EIGENVALUES
                          % USING THE GERSHGORING DISK
       STL(3)    .SAVE3;
       SETE      E.OR.-E;
       SETE1     E.AND.E;
       CLRA;
       LDS       $A;
%*********
%  THE RADIUS OF THE DISK ACC. TO GERSHGORIN-S TH" CONSISTS
%  OF THE SUM OF THE ABS. VALUE OF THE OFF-DIAG ELEMENTS
%  LOCATED IN ROW I FOR WHICH A(I,I)=EIGENVAL.I.E. THE ROWSUM
%  OF EACH ROW IS FOUND.  THE CENTER OF THE DISK IS THE EIGENVALUE
%*********
       LDL(1)    .SVLOOP;
%*********
%  FINDING THE SUM OF THE ROW IS EQUIVALENT TO FINDING THE SUM
%  OF THE COLUMS, SINCE THE MATRIX IS SYMMETRIC
%*********
GERS:  LDL(0)    .PEMODE;
       CCB(0)    0(1);
       LDEE1     $C0;
       LDL(2)    .ADRA;
       CADD(2)   $C1;
       LDA       0(2);
       SAP;
       ADRN      $S;
       LDS       $A;
       TXEFM(1)  .GERS;
       LDL(0)    .PEMODE;
       LDEE1     $C0;
       LDL(0)    .ADRG;
       STA       0(0);     % STORE BOUNDS ON EIGENVALUES
       LDL(3)    .SAVE3;
       EXCHL(3)  $ICR;
%*********
%*********************************************
JACOBI[ENTRY]:: FILL;     % SAVE $C0,$C1, ADB $D32-$D39
       CHWS      64;
       SETE      E.OR.-E;
       SETE1     E.AND.E;
       STL(3)    .RETUR;
       CLC(3);
       SLIT(3)   ADBSAV+8;
       STORE(3)  $C0;
       ALIT(3)   1;
       STORE(3)  $C1;
       CLC(3);
       SLIT(3)   ADBSAV;
       LIT(0)    1,7,0;
SA:    STORE(3)  $D32(0);
       ALIT(3)   1;
       TXEFM(0)  .SA;
       LIT(0)    1,4,0;
       CLC(3);
```

```
SA1:    LOAD(2)   $C3;        % $C2 SET IN CALL STMT FROM MAIN PROG
        CSHR(3)   6;
        STL(3)    $D34(0);  % ADDR OF DIAG,TEMP, PAIR AND EIGV MTXS
        ALIT(2)   1;
        TXEFM(0)  .SA1;
        LOAD(2)   $C3;
        LOAD(3)   $C3;
        STL(3)    .N;         % DIMENSION OF SYSTEM
        ALIT(2)   1;
        LOAD(2)   $C3;
        STL(3)    .ADRI;      % ADDR  OF SWEEP COUNT - TO BE OUTPUT
%
%  INITIALIZE CONSTANTS AND OTHER PARAMETERS
%
        CLC(0);
        STL(0)    .ZERO;
        STL(0)    .RSWTCH;
        STL(0)    .ICNT;
        STL(0)    .KCONV;
        LIT(0)    =1;
        STL(0)    .ONE;
        CADD(0)   $C0;
        STL(0)    .TWO;
        LIT(0)    =4;
        STL(0)    .BOUND;
        LIT(0)    =64;
        STL(0)    .SIXT4;
        CSUB(0)   .N;
        STL(0)    .ROUT;
        LDL(0)    .N;
        CSUB(0)   $D1;
        STL(0)    .N1;  % N MINUS 1
        LIT(0)    =0.1.0;
        CADD(0)   .N1;
        CROTL(0)  24;
        STL(0)    .SVLOOP;
        LDL(0)    .N;
        CADD(0)   $D1;
        CSHR(0)   1;
        STL(0)    .M;  % M=[ (N+1)/2 ]
        CSHL(0)   1;
        STL(0)    .M2;  % 2*M
        CSUB(0)   .N;
        STL(0)    .EVNODD;  % =0 IF N EVEN, =1 IF ODD
        LDL(0)    .M2;
        CSUB(0)   .ONE;
        STL(0)    .M2M1;
        CSHL(0)   1;
        STL(0)    .M4M2;     % 4*M-2
%
*  TURN ON FIRST N-1 PE-S
%
        LDA       PEN;
        LDL(0)    .N;
        IAL       $C0;
        SETC(0)   I;
        STL(0)    .PEMODE;
```

```
          LDEE1     $C0;
 %
 %   INITIALIZE EIGENVECTOR MATRIX
 %
          CLRA;
          LDL(0)    .SVLOOP;  % LOOP FROM 0 TO N-1
          LDL(1)    .ADRE;
 EVINIT:  STA       0(1);
          CADD(1)   .ONE;
          TXLTM(0)  .EVINIT;
          LDX       PEN;
          LIT(0)    =1.0;
          LDA       $C0;
          LDL(1)    .ADRE;
          STA       *0(1);
 %
 %   SET UP PAIRS (P,Q) FOR ANNIHILATION
 %
          LDL(1)    .PEMODE;
          LDEE1     $C1;      % TURN ON PE-S 0 TO N-1
 %
 %    SET UP LOOP COUNTER TO GO FROM 1 TO 2M-1
 %
          LIT(0)    =0,1,0;
          CADD(0)   .M2M1;
          CROTL(0)  24;       % LOOP FROM 1 TO 2M-1
          CADD(0)   $D1;
          LDL(1)    .M4M2;
          STL(1)    .INDEX;
 %
 SETUP:   EQLXFA(0) $D7.KNOTM;
          CADD(1)   .M2;
          CSUB(1)   .ONE;
          STL(1)    .INDEX;   % 6*M-3  RESET WHEN K=M
 KNOTM:   CSUB(1)   $C0;
          CSUB(1)   $C0;
          LDA       $C1;      % INDEX = 2*K
          SBM       PEN;      % INDEX = 2*K - Q
          LDL(2)    .M2M1;
          IAL       $C2;
          SETE      -I.AND.E;
          SETE1     E.AND.E;
          SBM       $C2;      % IF P  GEQ  2*M-1SUB 2*M-1
          LDL(2)    .PEMODE;  % TURN FIRST N-1 PE-S ON
          LDEE1     $C2;
          LDL(2)    .ADRP;    % GET ADDR OF PAIR MATRIX
          CADD(2)   $C0;      % GET I-TH ROW AND STORE P,Q
          STA       0(2);
          CLC(3);             % IF N IS ODD SKIP TO PQDONE
          EQLXFA(3) $D14.BUMPQ;
 %
          CSHL(2)   6;        % BACK TO CU ADDR
          CADD(2)   .N1;      % + (N-1)
          CSUB(2)   $C0;      %            -K
          LDL(3)    .N1;
          STORE(2)  $C3;      % STORE N-1 IN PE(N-1-K)
          CADD(2)   $C0;      % CU ADDR PROW + (N-1)
```

```
       CSUB(3)  $C0;       % N-1-K
       STORE(2) $C3;       % STORE N-1-K IN PE(N-1)
BUMPO: LDL(1)   .INDEX;
       TXLTM(0) .SETUP;
*
*****SWEEP LOOP - ONE SWEEP=2M-1 ITERATIONS
*           WHERE M=[(N&1)/2] , N=DIMENSION OF SYSTEM
SWEEP::LDL(0)   .ICNT;
       CADD(0)  .ONE;
       STL(0)   .ICNT;
       LDL(1)   .PEMODE;
       LDEE1    $C1;       % TURN ON PE-S 0 TO N-1
*
*   BEGIN ITERATION LOGIC
*
       LIT(0)   =0,1,0;
       CADD(0)  .M2M1;
       CROTL(0) 24;        % LOOP FROM 1 TO 2M-1
       CADD(0)  $D1;
LOOP:  STL(0)   .INLOOP;
       LDL(2)   .ADRP;     % GET ADDR OF PAIR MTX
       CADD(2)  $C0;       % PICK UP I-TH ROW
       LDA      0(2);      % GET PAIRS (P,Q) TO ANNIHILATE
       STA      PROW;
       CLC(3);
       SLIT(3)  =ANGLE;
       FXCHL(3) $ICR;
*
       LDL(2)   .ADRA;
       STL(2)   .ADR1;
       LDL(2)   .ADRB;
       STL(2)   .ADR2;
*
*  POST MULTIPLY MATRIX A BY THE TRANSFORMATION MATRIX
*  ACTUALLY PRE-MULTIPLY A BY TRANSPOSE OF TRANSFORM MTX
*  THEN TAKE TRANSPOSE OF PRODUCT
*
       CLC(3);
       SLIT(3)  =MULTPL;
       EXCHL(3) $ICR;
       LDL(0)   .ADRB;
       CADD(0)  .ONE;
       LIT(1)   =63;
       LDL(2)   .SVLOOP;
       CADD(2)  .ONE;      % LOOP FROM 1 TO N-1
       LDL(3)   .ONE;
       STL(3)   .RSWTCH;   % EFFECTIVE LEFT ROUTE
TRANS: LDA      0(0);      % ARRANGE MATRIXB TO YIELD TRANSPOSE
       STL(1)   .AMT;
       CLC(3);
       SLIT(3)  =ROUTE;
       FXCHL(3) $ICR;
       STA      0(0);
       ALIT(1)  =77777777:8;
       CADD(0)  .ONE;
       TXLTM(2) .TRANS;
*
```

```
*   MULTIPLY TRANSPOSE OF TRANSFORM MTX BY MATRIXB
*   YIELDING NEW MATRIXA WITH ALL A(P,Q) ANNIHILATED (HOPEFULLY)
          LDL(2)    .ADRB;
          STL(2)    .ADR1;
          LDL(2)    .ADRA;
          STL(2)    .ADR2;
          CLC(3);
          STL(3)    .RSWTCH; % RESET ROUTING SWITCH FOR RIGHT ROUTING
          SLIT(3)   =MULTPL;
          EXCHL(3)  $ICR;
          LDL(2)    .PEMODE;
          LDL(3)    .SAVI;
          COMPC(3);
          CAND(3)   $C2;
          LDEE1     $C3;
          LDL(0)    .ADRA;
          CLC(1);
          LDX       PROW;
          LDA       $C1;
          STA       *0(0);
          LDEE1     $C2;
*
*   UPDATE EIGENVECTOR MATRIX
*
          LDL(2)    .ADRE;
          STL(2)    .ADR1;
          LDL(2)    .ADRB;
          STL(2)    .ADR2;
          CLC(3);
          SLIT(3)   =MULTPL;
          EXCHL(3)  $ICR;
          LDL(0)    .ADRB;
          LDL(1)    .ADRE;
          LDL(2)    .SVLOOP; % LOOP FROM 0 TO N-1
COPY;     LDA       0(0);
          STA       0(1);      % COPY MATRIXB INTO EIGENVECTOR MATRIX
          CADD(0)   .ONE;
          CADD(1)   .ONE;
          TXLTM(2)  .COPY;
          LDL(0)    .INLOOP;
          TXLTM(0)  .LOOP;
% *************
          CLC(3);              % FIND CONVERGENCE FACTOR
          SLIT(3)   =ADDIT;
          EXCHL(3)  $ICR;
          ONESF(0)  .+1;
          SKIP      .EXIT;     % SUM OF OFF-DIAGONAL ELEMENTS=0
          LDL(0)    .KCONV;
          EQLXF(0)  $D0.CKICNT;
          LIT(2)    =.0000000000000001;
          LDA       $C2;
          LDL(2)    .RATIO;
          LDS       $C2;
          MLRN      $S;
          LDC(1)    $A;        % NOTE THAT ALL PE-S ARE ON (SEE ADDIT)
          STL(1)    .KCONV;    % KCONV = RATIO X 1.0E-16
CKICNT;LDL(0)      .ICNT ;
```

```
       LESSF(0) $D13,+1;  %DON-T CHECK RATIO WHILE ICNT < BOUND
       JUMP     SWEEP;
       LDL(1)   .RATIO;  %ICNT IS GEQ BOUND
       LDL(2)   .KCONV;
       LDA      $C1;
       IAL      $C2;      % SET I BITS IF RATIO<KCONV
       SETC(1)  I;
       ONEST(1) .+1;      % IF RATIO NOT LESS THAN KCONV CONTINUE
       JUMP     SWEEP;
%
% ******** END OF SWEEP LOGIC *************
%
EXIT:: CLC(3);
       SLIT(3)  =GERSH;
       EXCHL(3) $ICR;
       LDL(3)   .ICNT;
       LDL(1)   .ADRT;
       STORE(1) $C3;
EXIT1::CLC(3);
       SLIT(3)  ADRSAV;
       BIN(3)   $D32;
       CLC(3);
       SLIT(3)  ADRSAV+8;
       LOAD(3)  $C0;
       ALIT(3)  1;
       LOAD(3)  $C1;
       LDL(3)   .RETUR;  % RETURN TO MAIN PROGRAM
       EXCHL(3) $ICR;
       END JACOBI.
?END
```

UNCLASSIFIED

Security Classification

DOCUMENT CONTROL DATA - R & D

(Security classification of title, body of abstract and indexing annotation must be entered when the overall report is classified)

1. ORIGINATING ACTIVITY (Corporate author) Center for Advanced Computation University of Illinois at Urbana-Champaign Urbana, Illinois 61801	2a. REPORT SECURITY CLASSIFICATION UNCLASSIFIED 2b. GROUP

3. REPORT TITLE

A JACOBI ALGORITHM FOR ILLIAC IV

4. DESCRIPTIVE NOTES (Type of report and inclusive dates)

Research Report

5. AUTHOR(S) (First name, middle initial, last name)

Lawrence M. McDaniel

6. REPORT DATE	7a. TOTAL NO. OF PAGES	7b. NO. OF REFS
November 8, 1971 (revised June 7, 1972)	53	5

8a. CONTRACT OR GRANT NO. DAHC04 72-C-0001 b. PROJECT NO. ARPA Order 1899 c. d.	9a. ORIGINATOR'S REPORT NUMBER(S) CAC Document No. 21 9b. OTHER REPORT NO(S) (Any other numbers that may be assigned this report)

10. DISTRIBUTION STATEMENT

Copies may be obtained from the address given in (1) above.

11. SUPPLEMENTARY NOTES None	12. SPONSORING MILITARY ACTIVITY U.S. Army Research Office-Durham Duke Station Durham, North Carolina

13. ABSTRACT

This revised version of CAC Document #21 supercedes the document dated November 8, 1971.

Several methods have been proposed to enable the computation of eigenvalues and eigenvectors of large, real symmetric or complex Hermitian matrices on ILLIAC IV. One of the most effective methods in the utilization of parallel computations has proven to be a modified Jacobi algorithm. This document presents yet another modification which exploits the parallelism of ILLIAC IV to a greater extent than has been previously done. Flow charts and the assembly language (ASK) routine JACOBI are included in the report.

DD FORM 1 NOV 65 1473

UNCLASSIFIED

Security Classification

UNCLASSIFIED
Security Classification

14. KEY WORDS	LINK A		LINK B		LINK C	
	ROLE	WT	ROLE	WT	ROLE	WT
Matrix Algebra						

UNCLASSIFIED
Security Classification